



# **Aircraft Situation Display To Industry: Functional Description and Interface Control Document for the XML Version**



## **Version 1.2**

Contract Number: DTFAWA-04-C-00045  
CDRL Sequence No:

**September 7, 2007**

Prepared for:  
U.S. Federal Aviation Administration

Prepared by:  
Computer Sciences Corporation  
Federal Sector – Civil Group  
100 Decadon Drive  
Egg Harbor Township, NJ 08234



# **Aircraft Situation Display To Industry: Functional Description and Interface Control Document for the XML Version**

## **Version 1.2**

Contract Number: DTFAWA-04-C-00045  
CDRL:

**September 7, 2007**

Prepared for:  
**U.S. Federal Aviation Administration**

**Prepared by:**  
Computer Sciences Corporation  
Federal Sector – Civil Group  
100 Decadon Drive  
Egg Harbor Township, NJ 08234

### Document History Record

Release	Date	Comment
Initial Draft v0.8	March 15, 2007	Initial draft for internal review
Initial Draft v0.9	March 22, 2007	Initial draft – Incorporated comments from first review.
Draft v1.0	March 27, 2007	Draft for Distribution to Vendors/Clients
Draft v1.1	May 14, 2007	Incorporated comments against v1.0, replaced example of the preamble in Section 6 and updated Section 4.4 on security.
Draft v1.2	September 7, 2007	Revised Facility Identifier list in section 3.3 to reflect current operational values; added new section 3.5 for ARTCC codes Updated Contacts Section 1.2

## Table of Contents

1	Introduction .....	1-1
1.1	Background and Terminology .....	1-1
1.2	Contacts.....	1-2
1.3	Document Overview.....	1-3
1.4	Referenced Materials.....	1-3
2	Functional Overview .....	2-1
3	Content of the ASDI Data Stream Sent to the Vendors .....	3-1
3.1	Filtering Messages Out of the ASDI Data Stream.....	3-1
3.2	Removing Fields From the Allowed ASDI Messages.....	3-2
3.3	Facility Identifiers .....	3-2
3.4	Variations of TZ Messages .....	3-3
3.5	ARTCC Codes .....	3-4
4	Software Interface, Hardware Interface, and Security.....	4-1
4.1	Software Interface.....	4-1
4.1.1	Software Tips for Vendors .....	4-2
4.2	Needed Bandwidth.....	4-3
4.3	Summary of What Needs To Be Done if a Vendor Is To Receive the Feed.....	4-4
4.4	Security.....	4-4
4.5	Handling the London Data .....	4-6
4.5.1.1	What Users Can Get the London Data? .....	4-7
4.5.1.2	What Vendors Can Get the London Data? .....	4-7
4.5.1.3	How Can the London Data Be Identified?.....	4-8
4.6	Class One and Class Two Users .....	4-8
4.6.1.1	How Can a Vendor Get the Undelayed Feed.....	4-8
5	Structure of the XML Feed.....	5-1
5.1	Introduction .....	5-1
5.2	Structure of an ASDI Packet.....	5-1
5.3	Structure of the Header.....	5-1
5.4	Hints to Vendors on Processing Header and Payload .....	5-3
6	ASDI XML Message Formats .....	6-1
6.1	The Design of the ASDI Schema .....	6-1
6.1.1	Required ASDI Schema Definition Files and Locations .....	6-3
6.2	Structure of an ASDI XML Document .....	6-3
6.2.1	ASDI XML Preamble.....	6-4
6.2.2	ASDI XML Messages.....	6-4
6.3	Development, Test and Validation tools.....	6-4
6.4	Hints to Vendors to help parse ASDI XML documents.....	6-5
Appendix A	Sample ASDI Messages .....	A-1
A.1	ASDI Messages in the Original Format.....	A-1
A.2	ASDI Messages in XML Format.....	A-2
Appendix B	ASDI Schema Elements.....	B-1
B.1	Table Descriptions .....	B-1
B.1.2	NasCommon Elements.xsd .....	B-1

**List of Tables**

Table 4-1. Registration Message Format .....4-2

**List of Figures**

Figure 5-1. ASDI Data Packet.....5-1  
Figure 6-1. TFM-M XML Schema Design.....6-2

## **Foreword**

The Aircraft Situation Display to Industry (ASDI) subsystem of the Enhanced Traffic Management System (ETMS) allows near real-time air traffic data to be disseminated to members of the aviation industry. This document provides a functional description of the ASDI subsystem and gives the interface information that a vendor would need to connect to this subsystem or that a user would need to interpret the ASDI data.

The FAA is converting the ASDI feed to XML format, and this is the first version of this document that provides information about this XML version of the feed.

# 1 Introduction

---

## 1.1 Background and Terminology

In 1992 the Federal Aviation Administration (FAA) started providing airlines and other aviation-related organizations with access to near real-time air traffic data from the National Airspace System (NAS). This enabled airlines to use this more accurate positional information in their planning and decision making.

This original data feed was decommissioned in 1999 and was been replaced with an improved feed known as the Aircraft Situation Display to Industry (ASDI) feed. This feed, in turn, is being replaced in 2007 with a version of the feed that is in the XML format. This document explains the XML version of the feed.

The FAA's plan is that the legacy feed and the XML feed will be offered in parallel through 2007. By the beginning of 2008, however, all vendors are required to change to the XML feed; the legacy feed will be discontinued.

An organization that receives the ASDI feed directly from the FAA will be termed a *vendor*. An organization that uses the data is called a *user*. It is expected that a typical vendor will re-sell the data to users, though it might be the case that a vendor is also a user.

The purpose of this document is to provide vendors and users with the information that they need to receive and interpret the XML version of the ASDI feed. If a user purchases the feed from a vendor, it might be that this user will require additional information from the vendor.

## 1.2 Contacts

Throughout this document references are made to an FAA contact and a Volpe contact. As this is written, the contacts are as follows.

### **FAA contact information:**

Judy Morrill, (703) 326-3909, [judy.morrill@faa.gov](mailto:judy.morrill@faa.gov).  
Midori Tanino, (703)326-3937, [midori.tanino@faa.gov](mailto:midori.tanino@faa.gov).

### **Volpe contact information:**

To report a **data outage 24x7** call ETMS Operations Help Desk: (617) 494-2556.  
The Operations Help Desk staff can verify normal server functioning. Follow-up  
ALL calls with an email describing the problem to: [ASDI\\_Support@volpe.dot.gov](mailto:ASDI_Support@volpe.dot.gov)

To report all other issues, (for example, individual vendor connectivity issues and data questions), send an email to [ASDI\\_Support@volpe.dot.gov](mailto:ASDI_Support@volpe.dot.gov) or call Sean Sullivan at (617) 494-3486 during normal business hours.

### **Required Information from Vendors for Data Questions**

As described in the XASDI ICD, each XASDI data packet is comprised of two parts. The first is a fixed length header. The second is the compressed data payload. The header information being requested is for the header of the data packet that contains the message in question.

**The following table enumerates the data that vendors need to provide when making queries regarding XASDI data.** Given the new structure of the code and the corresponding new logging that is done by the XASDI Server, all of this information is necessary in order to identify the data in question.

Data	Description	Example
Type of Feed	Determines instance of the XASDI a vendor is connected to	Class 1
Timestamp in header of data packet	The timestamp in the data packet header	20070910205027 (i.e. 2007/09/10 20:50.27)
Packet sequence number of the data packet	The sequence number (in ASCII format) in the data packet header	00016
Message type	Message type in question. For example, if it was in a TZ message, that would be a trackInformation message type.	trackInformation
ACID	Flight's ACID	AAL123
XML tag and value	Each component of the XASDI is encapsulated in an XML container. Need to identify the tag and the associated value of the data in question.	<nxcn:speed> - tag 200 - value



Generally, a vendor should get in touch with the FAA contact to get information about the memorandum of agreement (MOA) that entitles the vendor to get the feed. After a vendor has signed that memorandum of agreement, it should get in touch with the Volpe contact to learn how to get connected, to ask technical questions about the feed, or to find out more about the documents listed in the next section. If necessary, Volpe will refer any questions to CSC, which is responsible for the format of the XML feed.

## 1.3 Document Overview

This ICD is organized into the main document sections detailing the features of the ASDI XML data feed and Appendices containing supporting information such as a Glossary and Acronyms.

- Section 1 – Introduction, defines the purpose of this ICD and provides a brief overview of the major sections of the document
- Section 2 – Functional Overview, describes a high-level functional description for the new ASDI XML data feed and explains some of the differences between the old and new feeds.
- Section 3 – Content of the ASDI Data Stream Sent to the Vendors, describes the data content, filtering, message types and other information contained in the ASDI data feed.
- Section 4 – Software Interface, Hardware Interface, and Security, describes the software and hardware interfaces and protocols between the ASDI server and client connections. It also includes specific sub-sections describing software tips for the vendors to follow, recommended bandwidth, a summary of what a vendor needs to do to connect, security requirements and an explanation about the different classes of users.
- Section 5 – Structure of the XML Feed, describes the new structure of an ASDI packet, including the header and compressed payload of data. These details for the uncompressed payload, that is the XML formats, are described later in Section 6.
- Section 6 – Message Formats
- Appendix A – Sample Messages
- Appendix B – Acronyms, defines the acronyms used in this ICD.

## 1.4 Referenced Materials

<http://www.fly.faa.gov/ASDI/asdi.html> - This web site contains current information about the ASDI feed. It also includes the XML schema definition files (XSD) and sample ASDI XML messages.

“Memorandum of Agreement for Industry Access to Aircraft Situation Display and National Airspace System Status Information Data,” June 1, 2006. This is the MOA that a vendor must sign in order to get access to the ASDI feed. The current version of this document can be found on the web site referenced above.

<http://nasdoc.faa.gov/> This web site contains FAA documentation on the National Aviation System. To find the current versions of the documents below, type “nas-md-311” in the search bar, click enter, and then click on HOST\_A5f1.5\_CPFSs. This will give a pick list of documents.

*National Airspace System En Route Configuration Management Document, Computer Program Functional Specifications: Remote Outputs, Model A5f1.5, NAS-MD-315, 4 October 2004, National En Route Automation Division, AOS-300, Federal Aviation Administration, William J. Hughes Technical Center, Atlantic City International Airport, New Jersey 08405. This document describes all the messages that the Host computer can generate; in particular, Section 3 documents messages destined for the Central Flow Automation Facility (CFAF), which are the messages sent to ETMS. From the pick list mentioned above, click on NAS-MD-315 to open this document.*

*National Airspace System En Route Configuration Management Document, Computer Program Functional Specifications: Route Conversion and Posting, Model A5f1.5, NAS-MD-312, 4 October 2004, National En Route Automation Division, AOS-300, Federal Aviation Administration, William J. Hughes Technical Center, Atlantic City International Airport, New Jersey 08405. This NAS specification describes the syntax of the field 10 (route of flight) that is used on the AF, FZ, and UZ messages. From the pick list mentioned above, click on NAS-MD-312 to open this document.*

*National Airspace System En Route Configuration Management Document, Computer Program Functional Specifications: Message Entry and Checking, Model A5f1.5, NAS-MD-311, 4 October 2004, National En Route Automation Division, AOS-300, Federal Aviation Administration, William J. Hughes Technical Center, Atlantic City International Airport, New Jersey 08405. Appendix E of this document is the official documentation of the content of each field in the NAS messages. From the pick list mentioned above, click on NAS-MD-311 to open this document.*

*Aircraft Situation Display to Industry: Functional Description and Interface Control Document, ver. 5.4, Volpe Center, report no. ASDI-FD-001, 15 November 2005. This document covers the legacy version of the ASDI feed and is replaced by the current document, which covers the XML version of the feed.*

## 2 Functional Overview

---

The heart of the ASDI feed is the ASDI Server. During the period of parallel operation of the old, legacy feed and the new, XML feed, the new, XML servers will run at the Traffic Flow Management (TFM) Production Center located at the William J. Hughes Technical Center in Atlantic City (WJHTC), New Jersey, while the old, legacy servers will continue to run at the Enhanced Traffic Management System (ETMS) Hubsite at the Volpe Center in Cambridge, MA. (The WJHTC will sometimes be referred to as the Tech Center.) Both versions of the ASDI servers share the following main features.

- Obtaining data: ETMS receives data from numerous air traffic control facilities (see the next section for a list). The ASDI Server receives all of these messages from ETMS.
- Processing data: The ASDI Server processes this data by filtering out messages that should not go out into the feed and by removing any fields that should not be sent out; see the next section for more details.
- Providing data: The ASDI Server will accept TCP/IP connections from approved vendors. When the connection is made, the ASDI Server sends to that vendor the messages that the vendor is entitled to receive. Once the vendor has received a message, it is up to the vendor to treat that message as it wishes (as long as it does not violate the memorandum of agreement it has signed with the FAA) and to distribute the data to its users; how the data is distributed by the vendors to the users is not the responsibility of the FAA and is not covered in this document.

The old and new servers differ in several ways.

- To connect to the old server, a vendor must install a dedicated line to the Volpe Center. To connect to the new server, a vendor must use a VPN over the Internet. An exception to this is that current vendors who transmit CDM data will be allowed to continue to use dedicated lines.
- The old server provides data in plain ASCII format, whereas the new server provides the data in XML format.
- The old server sends one message at a time, whereas the new server may batch messages in data packets of a configurable size. That is, one data packet may contain 32, 64, or any other amount of ASDI messages. Early bandwidth studies indicate that 32 messages per packet may be the most efficient use of available bandwidth. The initial number of messages per packet for the new XML feed will be determined prior to its availability at the WJHTC and will be communicated to the vendors.
- The old server sends uncompressed data, while the new server compresses the payload (messages) portion of a data packet; this means that the recipient of the data must decompress it. The structure of an ASDI data packet is further discussed in Section 5.

To get the ASDI XML feed, a vendor must sign the memorandum of agreement (see References) and set up a VPN over the Internet that provides sufficient bandwidth.

This document only covers the new, XML version of the feed. For documentation on the legacy feed, refer to *Aircraft Situation Display to Industry: Functional Description and Interface Control Document*, version 5.4, referenced in Section 1.4.

To provide reliability, the FAA provides two operational ASDI Servers. The normal condition is that both are running. If system maintenance is needed, one server will be taken down at a time so that at least one server is always running.

There are three classes of the ASDI feed. There are two factors used to distinguish these classes

- Does the feed contain the London data?
- Are the messages passed to the vendor immediately or with a delay?

The rationale for these distinctions is explained in the next section. This gives rise to three feeds.

Feed Number	Includes London Data?	Delayed?
1	Yes	No
2	No	Yes
3	No	No

Theoretically there could be a fourth class of the feed that contained the London data and that was delayed, but this class is not offered since there is no need for it. All vendors who connect to a server for the same class of the feed receive exactly the same data. If two connections are made to two different servers for the same class of the feed, the data will be almost the same but not exactly; for example, messages might be ordered slightly differently in the feeds from the two different servers.

## 3 Content of the ASDI Data Stream Sent to the Vendors

---

### 3.1 Filtering Messages Out of the ASDI Data Stream

The ASDI Server registers with ETMS, which provides the ASDI server with the following messages.

- All of the NAS messages received by ETMS.
- All of the Position Update messages provided by the Surface Movement Advisor program to ETMS.
- All of the oceanic messages generated by ETMS.
- All of the RT messages generated by ETMS.

In the old, legacy feed these messages were in ASCII, while in the new, XML feed they are in XML.

Before sending these messages over the ASDI feed, the ASDI server filters the messages in various ways.

- Sensitive message types are filtered out. The only message types currently allowed into the ASDI data stream are the following, where the message identifier in the legacy feed is given in brackets:
  - Flight plan [FZ]
  - Flight plan amendment [AF]
  - Departure [DZ]
  - Track information, i.e., a position report [TZ]
  - Boundary crossing [UZ]
  - Flight management information [RT]
  - Oceanic report [TO]
  - Arrival [AZ]
  - Flight plan cancellation [RZ]
  - Heartbeat [HB] (Note: In the legacy feed, this is a message. In the XML feed this is represented by in the data header. See Section 5 for details).See Section 6 for an explanation of these messages. An example of the type of message that is filtered out of the feed is the beacon code [BZ] message.
- Messages on sensitive flights are filtered out. For example, messages on military flights are not allowed into the ASDI data stream.
- Messages from certain facilities are filtered out. For example, currently messages from Mexico are filtered out of the feed. Currently messages are included from facilities in the United States, Canada, and the United Kingdom; see Section 3.3 for a list of the facilities that send data to ETMS. (There are two versions of the feed; one contains messages from the United Kingdom and the other doesn't, as explained below.)

It should be noted that the rules for filtering out sensitive flights are conservative in the sense that some flights that are not truly sensitive are filtered out; for example, most messages on GA flights registered in countries other than the U.S. and Canada are filtered out. If a vendor thinks that certain flights that are filtered from the feed should be included, that vendor should get in touch with the FAA contact; the ASDI server can be configured to allow data on these flights into the feed.

## 3.2 Removing Fields From the Allowed ASDI Messages

If a message passes the filtering rules and is eligible to be sent out over the ASDI feed, the ASDI Server removes field 11 (remarks), if present, from this message before it is sent out.

## 3.3 Facility Identifiers

The following are the facility identifiers used to label the origin (source facility) of NAS messages. Note that there is a separate identifier used for ARTCC centers (see section 3.5 below), corresponding within the XML feed to departure airport facility id and centers within the flight traversal data.

The first column contains the one character code used in the raw data. Users of the XML ASDI feed do not see this value. The second column shows the four character identifier used in the ASDI feed; in some cases, the “K” might not appear.

Following the identifier is a brief description of the facility that generated the message. Note that ‘X’ and ‘x’ do not represent a NAS facility; rather, these codes indicate messages that are generated by ETMS, i.e., RT and TO messages. The ASDI feed represents each facility with a four character identifier, which is right justified and filled with leading blanks if necessary. These IDs, especially the TRACON and foreign IDs, are tentative. In some cases, facilities that provide multiple data sources will have the same facility identifier (i.e., ZAN EARTS and ZAN OCS). ODAPS stands for the Oceanic Display and Planning System, which provides NAS messages other than TZs on oceanic flights; position updates on oceanic flights are in the TO message.

A	KZAB	Albuquerque Center
B	KZBW	Boston Center
C	KZOB	Cleveland Center
D	KZDV	Denver Center
E	PZAN	Anchorage Center
\	PZAN	Anchorage Center
F	KZFW	Fort Worth Center
G	KZAU	Chicago Center
H	KZHU	Houston Center
I	KZID	Indianapolis Center
J	KZJX	Jacksonville Center
K	KZKC	Kansas City Center
L	KZLA	Los Angeles Center
M	KZME	Memphis Center
N	KZNY	New York Center
O	KZOA	Oakland Center
P	KZMP	Minneapolis Center
Q	PZHN	Honolulu, Hawaii (NAS TZ only)
R	KZMA	Miami Center

S	KZSE	Seattle Center
T	KZTL	Atlanta Center
U	KZLC	Salt Lake Center
V	PZLB	American Samoa (No data)
W	KZDC	Washington (D.C.) Center
X	ETMS	RT message generated by ETMS
Y	TZSU	San Juan, Puerto Rico
b	KF90	Boston TRACON (TZ only)
c	KCVG	Cincinnati TRACON (TZ only)
d	KDEN	Denver TRACON (NAS TZ only)
f	KDFW	Dallas/Fort Worth TRACON (NAS TZ only)
g	KC93	Chicago TRACON (TZ only)
h	KIND	Indianapolis TRACON (TZ only)
i	KSDF	Louisville TRACON (TZ only)
j	KMCO	Orlando TRACON (TZ only)
k	KGW	Gateway (St. Louis) TRACON (NAS TZ only)
l	KPHX	Phoenix TRACON (TZ only)
m	KMES	Memphis Tracon (SMA TZ only)
n	KN90	New York TRACON (NAS TZ only)
o	KOOA	ODAPS Oakland (AF, AZ, DZ, FZ, RZ, UZ only)
p	KPHL	Philadelphia TRACON (TZ only)
q	PZUA	Guam
r	PTPA	Tampa
t	KA80	Atlanta TRACON (NAS TZ only)
u	KNCT	Northern Cal TRACON (NAS TZ only)
v	KONY	ODAPS New York (AF, AZ, DZ, FZ, RZ, UZ only)
w	KPCT	Potomac TRACON (NAS TZ only) (No data)
x	ETMS	TO message generated by ETMS
z	KMSP	Minneapolis TRACON (NAS TZ only)
1	KSCT	Southern California TRACON (NAS TZ only)
0	YOWT	CANADA OTHER
2	CCZX	GANDER
3	CCZM	MONCTON
4	CCZU	MONTREAL
5	CCZY	TORONTO
6	CCZW	WINNIPEG
7	CCZE	EDMONTON
8	CCZV	VANCOUVER
<	LLON	LONDON
(	KA8S	Atlanta TRACON (SMA TZ only)
)	KF9S	Boston TRACON (SMA TZ only)
#	KC9S	Chicago TRACON (SMA TZ only)
+	KCLS	Charlotte TRACON (SMA TZ only)
\$	KDFS	Dallas-Ft. Worth TRACON (SMA TZ only)
@	KDTS	Detroit TRACON (SMA TZ only)
-	KSDS	Louisville TRACON (SMA TZ only)
&	KMSS	Minneapolis TRACON (SMA TZ only)
%	KN9S	New York TRACON (SMA TZ only)
=	KPHS	Philadelphia TRACON (SMA TZ only)
[	KPIS	Pittsburgh TRACON (SMA TZ only)
*	KGTS	St. Louis TRACON (SMA TZ only)

### 3.4 Variations of TZ Messages

The ASDI feed has four different sources of TZ messages, which give the current position of a flight.

- All of the ARTCCs send NAS TZs (as well as many other message types) directly to ETMS. Each of these ARTCCs has its own facility identifier.

- The ARTSIIE TRACONS send TZs directly to ETMS, and they send no other message type. Each of these ARTSIIE TRACONS has its own facility identifier.
- All of the TRACONS that are not ARTSIIE TRACONS send TZs through the associated ARTCCs to ETMS, and they send no other message types. The messages from these TRACONS have the facility identifier of the ARTCC through which the message was sent.
- The TRACONS that are SMA sites send messages directly to ETMS that are translated into TZs. The SMA sites send no other message types. Each SMA site has its own facility identifier that is used only for the SMA data.

The first three of these sources send TZs on each flight at roughly one minute intervals. In contrast, the SMA TZs are sent at roughly twenty second intervals.

It should be emphasized that since the SMA data was included in the ASDI feed in May of 2004, there have been two streams of TZs for each of the TRACONS that are SMA sites. For example, the following two streams of data from the Atlanta TRACON are included in the ASDI feed.

- A stream with 60 second updates with the identifier KA80. This is the stream that has long been included in the feed and is contributed by the flow of NAS messages to ETMS.
- A stream with 20 second updates with identifier KA8(. This is the new stream contributed by SMA.

In other words, once the SMA data was added to the ASDI feed, the feed continued to receive the data it has always gotten, plus it received additional data that duplicates the traditional data but that provides faster updates.

### 3.5 ARTCC Codes

The Facility Identifiers listed above in section 3.3 correspond to the logical locations which send NAS messages. The ARTCC codes listed in this section correspond to actual physical areas containing airports and airspaces, and are associated with the departure airport, as well as the centers the flight traverses. Many of the one character codes in the two tables are the same, but there are some in each table which are not in the other. For the XML ASDI feed, the one character internal ARTCC codes are translated into the associated 3 character code within the tags for `departurePointfacilityId`, and centers within the `flightTraversalData`.

Note that for international flights, the foreign centers may be artificially generated as one of the listed codes for London/Europe, South America, Pacific/Australia, etc. since the actual foreign centers may not be known or recognized.

The first column contains the three character code used in XML ASDI feed. The second column contains the one character raw data. Users of the XML ASDI feed do not see this value. The third column is a brief description of the ARTCC center corresponding to the code.



CZE	7	Edmonton, Alberta
CZM	3	Moncton, New Brunswick
CZU	4	Montreal, Quebec
CZV	8	Vancouver, British Columbia
CZW	6	Winnipeg, Manitoba
CZQ	2	Gander Domestic, Newfoundland
CZY	5	Toronto, Ontario
CZX	9	Gander Oceanic, Newfoundland
ZAB	A	Albuquerque, New Mexico
ZAN	E	Anchorage, Alaska
ZAU	G	Aurora, Illinois (Chicago)
ZBW	B	Boston, Massachusetts
ZDC	W	Washington, District of Columbia
ZDV	D	Denver, Colorado
ZEU	<	London (and Europe and India)
ZFW	F	Dallas/Fort Worth, Texas
ZHN	Q	Honolulu, Hawaii
ZHU	H	Houston, Texas
ZID	I	Indianapolis, Indiana
ZJX	J	Jacksonville, Florida
ZKC	K	Kansas City, Missouri
ZLA	L	Los Angeles, California
ZLB	V	American Samoa
ZLC	U	Salt Lake City, Utah
ZMA	R	Miami, Florida
ZME	M	Memphis, Tennessee
ZMP	P	Minneapolis, Minnesota
ZMX	!	Mexico
ZNY	N	New York, New York (Inland Section)
ZOA	O	Oakland, California
ZOB	C	Oberlin, Ohio (Cleveland)
ZPA	>	Pacific and Australia
ZSA	^	South America
ZSE	S	Seattle, Washington
ZSU	Y	San Juan, Puerto Rico
ZTL	T	Atlanta, Georgia
ZUA	q	GUAM
ZCA	?	Central America
ZCX	/	Chile
ZCO	~	Colombia

## 4 Software Interface, Hardware Interface, and Security

---

### 4.1 Software Interface

The interface between the ASDI server running at the William J Hughes Technical Center (WJHTC) and the client software running on the vendor's premises is a standard TCP/IP socket. The redundant ASDI servers each listen separately on assigned IP addresses and port numbers for clients that request to be registered to receive the data. After the vendor signs the ASDI memorandum of agreement with the FAA, it will be given these IP addresses and port numbers that it can use to connect of the ASDI servers. The IP addresses from which the vendor will connect must be provided to the Volpe Center so that the Tech Center can configure to allow clients from those addresses to connect to the appropriate servers.

An ASDI server sends identical data to all connected clients. The ASDI server buffers packets for each connected client. The ASDI server will discard data if the client's buffer overflows; this buffer holds a configurable number of packets per client (currently 16, but this could change). Any write to a client that results in an error condition will cause the ASDI server to terminate the connection. If the connection is terminated, the client must re-establish a connection if it is to receive data. No data will be buffered for a disconnected client. A 'port filled' or 'no room in port' condition is not considered to be an error; if either of these conditions is returned when the ASDI server attempts to write to a socket, then this data is buffered, and later an attempt is made to re-transmit.

The ASDI server uses non-blocking write operations to its clients. The ASDI server sends all packets in sequence. The standard interpretation of any missing packets is that they resulted from a buffer overflow forced by (1) the client's not reading the messages fast enough or (2) the client's communication link not being able to accept the ASDI data feed fast enough.

After a vendor opens a socket, it is given 60 seconds to send a vendor name and password to the ASDI server. The message that contains the vendor name and password is called a registration message; once the ASDI server has received a valid registration message, the client is said to be registered. No data is sent until the ASDI server receives a valid registration message. If 60 seconds after the socket connection is made the ASDI server has not received a legal registration message, the ASDI server will close this socket. The table below specifies the format of the registration message, which contains a vendor name and password. Each element of this message may be delimited by one or more spaces. Spaces directly after an = sign are discarded and are not considered part of the vendor name or password.

**Table 4-1. Registration Message Format**

<b>ID</b>	<i>Keyword specifying id field.</i>
<b>=</b>	<i>Field assignment character.</i>
<b>&lt;vendor name string&gt;</b>	<i>1 - 80 character alphanumeric string. Chosen by the vendor. May include embedded spaces.</i>
<b>,</b>	<i>Field separator character.</i>
<b>PASSWORD</b>	<i>Keyword specifying password field.</i>
<b>=</b>	<i>Field assignment character.</i>
<b>&lt;vendor password string&gt;</b>	<i>1 – 12 character alphanumeric string. Chosen by the vendor.</i>

Following is an example of a registration message:

ID = I AM A VENDOR , PASSWORD = mypassword

A vendor can choose any number of vendor name/password pairs; the vendors are encouraged to have back-up pairs. They should be communicated to the Volpe contact. It is required that the vendor name includes in some way the name of the vendor; this will make it easier for support personnel to monitor the system and deal with problems.

A vendor will be allowed to connect once to each ASDI server with each password. It is expected that under normal conditions a vendor will have not more than two active connections to the ASDI servers, where one is used for production and one for testing; it is expected that additional connections will be made only in exceptional cases, e.g., if extra testing is needed.

The vendor name and password are needed when a vendor connects to an ASDI Server. How a user connects to a vendor to get the feed is beyond the scope of this document; a user should contact its vendor to learn how to connect to receive the feed from the vendor.

### 4.1.1 Software Tips for Vendors

To provide redundancy, the WJHTC runs two ASDI servers and vendors will be given an IP address for each server. The FAA strongly recommends that a vendor write its client software so that, if it is unable to connect and get data from one of the IP addresses, it will automatically try the other address, and that it keeps trying these two addresses until it makes a connection. By cycling through the addresses in this way, a vendor can minimize down time.

The FAA's assumption is that if there is at least one IP address providing data, the vendors will be able to find it. Therefore, if the FAA needs to take down the servers

to do maintenance, the FAA will take down one at a time. While the first server is down, vendors can connect to the second; when the first is back in service, the second will be taken down. In a situation like this, no planned outage will be announced since it is assumed that the vendor client software can deal with it.

In short, it is the FAA's responsibility to make sure that at least one of the two addresses is providing data; it is the vendor's responsibility to find that address and connect to it. (As a matter of fact, experience shows that both addresses are operating normally much more than 99 percent of the time, and it is extremely rare for neither address to be available.)

The FAA strongly recommends that vendor software discard any unknown XML tags. There has been discussion of adding new data to the feed. If the FAA decides to add additional data, it is required to give at least 60 days notice of a change such as this. Nevertheless, if a vendor's client software discards unknown XML tags, the vendor in this situation can avoid having to develop software against the clock.

## 4.2 Needed Bandwidth

It is the responsibility of the vendor to make sure that its connection to the Internet has sufficient bandwidth to carry the feed. A frequently asked question asked by vendors is: How much bandwidth is required? Unfortunately, it is not possible to give an entirely satisfactory answer to this question. The following are, however, guidelines.

- As this is written, a 128 kilobits per second (Kbps) line can just barely handle a feed. To prevent this feed from backing up, however, and to allow for future growth of the feed, a minimum bandwidth of 256Kbps is recommended.
- It will sometimes be the case, however, that a vendor will want to get two copies of the feed, e.g., a primary feed and a back-up feed, or a production feed and a test feed. For example, when a change is being made to the feed, sometimes the FAA will make the new feed available on a test basis so that the vendors can test with it before it becomes the production feed. A prudent vendor would want to have enough capacity to connect for both a production feed and a test feed. This means that a vendor might well want to have the capacity for two feeds, which would be 512Kbps. A vendor might want more bandwidth if it needs to get more than one class of the feed.
- A larger capacity might be needed if the vendor plans to use this line for additional types of data. For example, currently Class One vendors have the option of getting digital Runway Visual Range data, as well as additional types of data that are available under the TFM Data to Industry Program.
- The FAA has plans to increase the amount of data in the feed. No timetable has been announced, but a prudent vendor would purchase a little extra capacity so that it is ready for future additions.

In summary, a prudent vendor will want to be ready for the expected increase in the quantity of data with a line of 256Kbps, 512Kbps, or perhaps more. It should be stressed, however, that the decision on what bandwidth to buy is entirely the responsibility of the vendor; the remarks above are designed only to provide some facts to the vendor who is facing this decision.

### **4.3 Summary of What Needs To Be Done if a Vendor Is To Receive the Feed**

The steps vendors must take to connect to an ASDI Server and get the feed are as follows:

- The vendor must send 2 signed copies of the ASDI memorandum of agreement to the FAA point of contact, along with the completed Vendor Information Form. Both documents can be found on the web site referred to in Section 1.4.”
- The FAA point of contact must inform the Volpe contact that a vendor has signed the documents required in the first step.
- The vendor must work with Volpe to set up a VPN over the Internet.
- The vendor must provide the Volpe contact with the IP address(es) of its VPN Gateway.
- The vendor must provide the Volpe contact with the IP address(es) of the clients that will connect to the ASDI Servers.
- The vendor must provide the Volpe contact with the vendor name/password pairs that it will use.
- The Volpe contact must provide the vendor with the IP addresses/ports on which the ASDI servers listen for connections.

If a vendor wants to receive the London data, it must go through additional steps spelled out in Section 4.6.

### **4.4 Security**

This section deals with general and application security issues.

Please note that the VPN and network security requirements are specified in another document, “XML ASDI via the Internet: VPN End-User Requirements and Guidelines”, which is available on the FAA web site. Vendors must meet all requirements specified in that document, as well as those detailed below.

One of the general requirements of the ASDI program is that security be maintained. There are two phases involved in maintaining security:

1. Vendors only engage in permitted activities.
2. Vendors take measures to detect and repel any prohibited activities.

The primary purpose of this section is to spell out which activities are permitted and which are prohibited so that vendors will not be in doubt about what is permitted. The secondary purpose is to state who is responsible when an prohibited action takes place.

## Overview

The general principle is: **Any activity that is not explicitly permitted is prohibited.** As this document is written, only the following activities are permitted, and these are only permitted from an approved IP address.

A vendor is only permitted to:

1. Register with the ASDI server to receive ASDI data
2. Manually telnet to addresses and ports to test connectivity
3. Register to get the digital feed of RVR data, if this vendor is eligible under section 11 of the ASDI MOA.”
4. Access the TFM Data to Industry Server

Other parties, e.g., organizations that only receive ASDI data from a vendor, are prohibited from all activity, unless this activity is approved in writing by the FAA contact.

It is expected that this list of permitted activities will change over time. Any changes to the list of permitted activities will be documented.

## Detailed Guidelines

The following guidelines flesh out the general principle above by pointing out some of the prohibited activities and by specifying who is responsible for a violation of these guidelines. This list is not meant to be an exhaustive list of the security violations; it merely adds detail about some of the more likely violations.

**If a prohibited activity is launched from an organization’s IP domain, then that organization is responsible for that violation.**

For example, if a disgruntled employee engages in a prohibited activity from an authorized IP address, then the vendor is responsible for this violation. For another example, if a hacker uses the Internet to break into a vendor’s network and then engages in prohibited activity, then this vendor is responsible for this attack.

**A vendor shall only allow the Tech Center to be accessed by parties operating from an approved IP address.**

That is, the vendor is responsible for providing network security.

**Any attempt to access an IP address other than those that are authorized shall be prohibited.**

At the moment, the only authorized IP addresses are those of the various servers specified above. In addition, there might be IP addresses that the FAA does not see, e.g., the IP address that an airline uses to register with a vendor to receive ASDI data.

**Any attempt to access a port other than a prescribed, published port shall be prohibited.**

For example, if someone tries to access an unused port on the ASDI Server, this will be construed to be an attack.

**IP spoofing shall be prohibited.**

That is, it is prohibited for one party to attempt to use another's IP address.

**An organization shall only capture or read data that it is explicitly authorized to look at.**

A vendor is prohibited from using a packet sniffer to capture data that it is not authorized to access.

**A password is associated with a set of IP addresses. Use of that password from some other IP address is prohibited.**

This means that it is not allowed for one vendor to use another's password.

### **Summary**

These guidelines are not exhaustive in that they do not spell out all of the prohibited activities; they only deal with the more important or the more likely prohibited activities. Again, the general principle is: **Any activity that is not explicitly permitted is prohibited.** If an ASDI vendor engages in a prohibited activity, then under section 12 of the ASDI MOA, the FAA has the right to cut off that vendor's access to the ASDI feed and any other feeds that the vendor is receiving.

## **4.5 Handling the London Data**

British flight data sent to ETMS is called the London data. The British authorities have placed on the FAA the requirement that the distribution of this data be restricted. The Tech Center operates ASDI servers that provide a feed that contains the London data, and it also operates ASDI servers that provide a feed from which the London data has been removed. This section discusses this requirement and its implications for ASDI vendors.

#### 4.5.1.1 What Users Can Get the London Data?

The British require that a user only receive the London data if it satisfies one or both of the following conditions.

- It is an air carrier. This includes cargo carriers.
- It owns and operates aircraft in Europe, and, therefore, pays landing fees and air traffic control fees in Europe.

If a user satisfies at least one of these conditions, it is said to be an *approved recipient* of the London data. The requirement imposed by the British, then, can be stated as saying that only approved recipients can receive the data. If a user does not fulfill at least one of these two conditions, then that user is not allowed to receive any of the London data.

#### 4.5.1.2 What Vendors Can Get the London Data?

By default, a vendor does not get the London data. That is, unless a vendor has been given explicit permission to get the London data, the FAA will only allow that vendor to register to get the feed that does not contain the London data. If a vendor wants to get approval to register for the feed that contains the London data, it must complete three steps.

- The vendor must certify in writing to the FAA contact that it has at least one customer that is an approved recipient of the London data.
- The vendor must submit to the FAA contact a plan that spells out how it will handle the London data so that it can guarantee that the London data will only go to approved recipients. The vendor must receive approval of this plan from the FAA contact.
- The vendor must certify to the FAA contact that it has implemented this approved plan.

Once a vendor has completed these three steps, the FAA will allow it to register to receive the London data. If a vendor later decides that it would like to change its plan for dealing with the London data, it can submit a revised plan to the FAA. Once the FAA approves a revised plan, it then becomes the plan that the vendor is required to follow.

If at any time the FAA discovers that a vendor is sending London data to an organization that is not an approved recipient, then the FAA has the option of immediately cutting off that vendor's access to the feed that contains London data; that vendor would then only be allowed to access the feed that does not contain the London data.



### **4.5.1.3 How Can the London Data Be Identified?**

Each message on the ASDI feed has four characters that indicate the facility that generated the message. This facility identifier can be found by looking at the sourceFacility attribute of the <asdiMessage> Element. If this facility identifier is 'LLON', then this message is from London. If this facility identifier is anything else, then this message is not from London.

## **4.6 Class One and Class Two Users**

When the ASDI feed was first established, the data was sent out to all vendors and users in near real-time, i.e., with a delay that in practice was a few seconds. The FAA has decided that for security reasons this near real-time feed should only go to those organizations with an established flight dispatch or planning function that requires near real-time data; these organizations are called Class One users. All other users are Class Two users, and these users would receive the feed with a delay. Examples of Class One users would be air carriers, regional air carriers, air taxis, and any organization that provides dispatch or tracking functions for aircraft owners, flight operation centers, government users, and professional flight planning service organizations. Examples of Class Two users would be most general aviation and non-aviation related industries. For a definition of Class One and Class Two users, see sections 5.2 and 5.3 of the ASDI MOA referenced in section 1.4.

A vendor that only has Class Two users will only be allowed to get a delayed feed. A vendor that has both Class One and Class Two users will be allowed to get both feeds; these vendors are obligated by the Memorandum to Agreement signed with the FAA to provide the undelayed feed only to Class One users. The FAA requires that all recipients of Class One data be audited annually at their own expense.

The FAA retains the option of changing the amount of delay.

### **4.6.1.1 How Can a Vendor Get the Undelayed Feed**

By default, a vendor receives the delayed feed. That is, unless a vendor has been given explicit permission to get the undelayed feed, the FAA will only allow that vendor to register to get the delayed feed. If a vendor has a Class One customer and wants to get approval to register for the undelayed feed, it must complete three steps.

- The vendor must certify in writing to the FAA contact that it has at least one customer that is a Class One user.
- The vendor must submit to the FAA contact a plan that spells out how it will handle the undelayed feed so that it can guarantee that the undelayed data will

only go to Class One users. The vendor must receive approval of this plan from the FAA contact.

- The vendor must certify to the FAA contact that it has implemented this approved plan.

Once a vendor has completed these three steps, the FAA will allow it to register to receive the undelayed feed. If a vendor later decides that it would like to change its plan for dealing with the undelayed data, it can submit a revised plan to the FAA. Once the FAA approves a revised plan, it then becomes the plan that the vendor is required to follow.

If at any time the FAA discovers that a vendor is sending undelayed data to an organization that is not a Class One user, then the FAA has the option of immediately cutting off that vendor's access to the feed that contains undelayed data; that vendor would then only be allowed to access the delayed feed.

## 5 Structure of the XML Feed

### 5.1 Introduction

The structure of the new, XML feed differs from that of the old, legacy feed. ETMS receives a stream of flight data messages, and the legacy feed sends each message individually and uncompressed into the ASDI feed. This method will not work with the XML version of the feed because it would use excessive bandwidth. The XML version instead accumulates a number of messages, compresses them, puts a header on this compressed data, and then sends out this packet that consists of a header and compressed data. This structure will now be explained.

### 5.2 Structure of an ASDI Packet

Each data packet that is transmitted to the ASDI vendor will be comprised of a fixed-length header, followed by a variable-sized payload. Following is a description of this format.

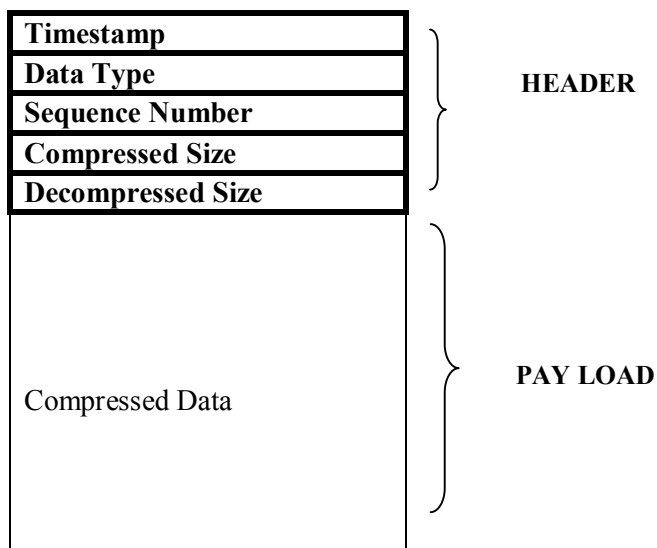


Figure 5-1. ASDI Data Packet

### 5.3 Structure of the Header

The header consists of the five fields shown in the figure above. The following data structure, written in C, describes the components of this header.

```
struct typedef xml_header_t
{
    char timestamp[16];
    int data_type;
```

```
    int sequence_number;  
    int compressed_size;  
    int decompressed_size;  
} xml_header_t;
```

The timestamp header component is 16 bytes long, and the other four fields are 4 bytes long. Therefore, the header is always 32 bytes long. The vendor's client software, therefore, always needs to read exactly 32 bytes to read a header.

The **timestamp header** component is a null terminated ASCII character array in `yyyymmddhhmmss` format that shows when the packet was transmitted by the ASDI server. 16 bytes are allocated to the timestamp to assure that all header components are word-aligned. The first fourteen bytes of this field is the timestamp; the fifteenth byte is a null character; the last byte is an blank character.

All other components of the header are binary data, represented by 4-byte integer values. These components are converted to network format before being sent to the vendors.

There are two types of packets that are sent. The most common packet type contains flight data messages in XML format. The other packet type is a heartbeat message that indicates to the vendor client software that the ASDI Server is alive. This allows the client software to distinguish the case where the server is down from the case where the server is up but receiving no data from ETMS. The vendor software can use this to decide when to switch to the other ASDI Server. The **Data Type** field indicates which type of packet is being sent by using the following values in the `data_type` field:

```
#define HeartBeatDataType 1  
#define XMLDataType 2
```

Each packet is given a **sequence number**. The vendor software can, if desired, keep track of these sequence numbers to see if all packets are being received. The valid range for the sequence number is 0 through 100000, inclusive. However, note that the sequence number is set to 0 only at initial program startup. When the sequence number value reaches 100000, the sequence number is reset to 1. This provides a method for the vendor software to detect when the server has restarted.

As mentioned, this structure will be converted to network format before it is sent to the ASDI vendors.

The **compressed size** gives the size of the compressed data in the payload. This tells the client software how big a buffer is needed to read the payload. The ASDI Server performs in-memory compression using the standard zlib compression library. These are the same compression routines on which the gzip utilities are built. Please refer to [www.zlib.net](http://www.zlib.net) and [www.gzip.com](http://www.gzip.com) for details.

The **decompressed size** gives the size of the data in the payload after it is decompressed. This tells the client software how big a buffer is needed to hold the payload after it is decompressed.

Once the payload is decompressed, the data is in XML. The format of the preamble and messages in the decompressed payload is covered in Section 6.

## 5.4 Hints to Vendors on Processing Header and Payload

Upon receipt, the ASDI vendor will need to process the header and convert it back to host format. The following is a sample of C code that could be used to perform this conversion. This sample presumes that the data has already been read into buff from the data stream.

```
format_header(char *buff, xml_header_t *header)
{
    char *ptr;

    /* Unpack data stream */
    ptr = buff;
    strcpy(header->timestamp, ptr);
    ptr += sizeof(header->timestamp);
    memcpy(header->data_type, sizeof(header->data_type));
    ptr += sizeof(header->data_type);
    memcpy(header->sequence_number, sizeof(header->sequence_number));
    ptr += sizeof(header->sequence_number);
    memcpy(header->compressed_size, sizeof(header->compressed_size));
    ptr += sizeof(header->compressed_size);
    memcpy(header->decompressed_size, sizeof(header->decompressed_size));

    /* Convert to host format */
    header->data_type = ntohl(header->data_type);
    header->sequence_number = ntohl(header->sequence_number);
    header->compressed_size = ntohl(header->compressed_size);
    header->decompressed_size = ntohl(header->decompressed_size);
}
```

Once the header is processed, the vendor would then read the compressed payload, uncompress it, and process the XML data. The sequence number in the header would be used to determine if any data packets were lost during processing.

## 6 ASDI XML Message Formats

---

As mentioned in previous sections the new format for ASDI messages is XML. Providing this data in XML format has several advantages to the clients of this feed, mainly validated machine readable data. XML is a well established industry standard and as such tutorials, reference material, documentation, development and test tools and support are readily available.

Transitioning to the ETMS ASDI XML feed requires changes to current ASDI vendor/client software that reads, parses and processes the data based on the old formats. This section mainly addresses the structure of the XML schemas and formats of the messages populated with those schemas. And to a certain extent will provide a basis of understanding for assisting vendors and clients in making their software changes.

The process of connecting to the feed and the structure the data packets sent over the feed were discussed in previous sections.

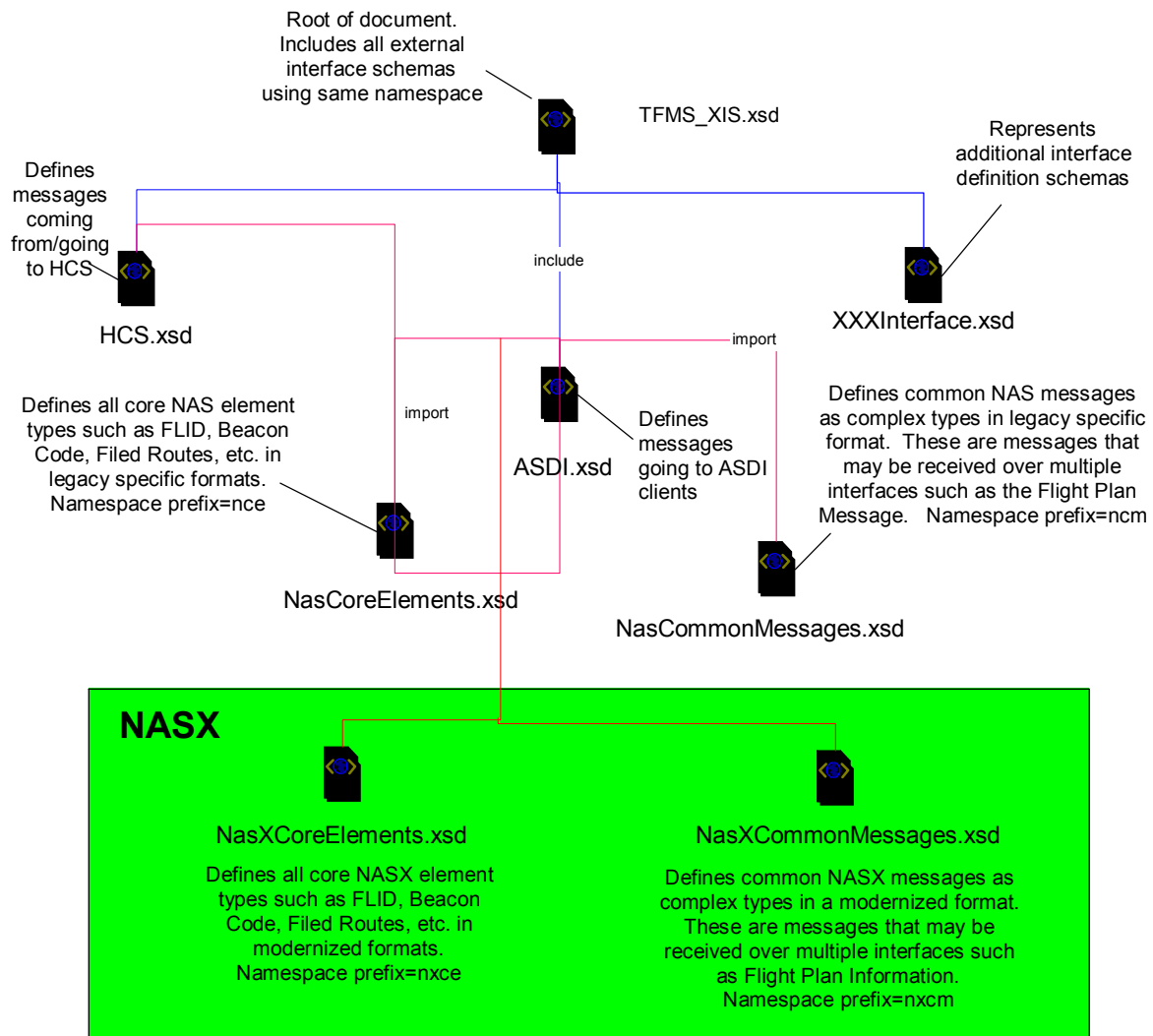
### 6.1 The Design of the ASDI Schema

The ASDI schema was designed and developed as part of the TFM Modernization (TFM-M) effort. The approach taken was to develop a comprehensive XML schema that is descriptive and extensible and to include all data and types that will eventually be available from the TFM-M system. The schema design used a hierarchal approach starting with a root schema representing the namespaces (schemas) needed to interpret the data content of the XML feed. Lower level schemas are then imported into this root level schema as needed by specific XML data feeds. The next level of the schema design hierarchy contains the specific interfaces or data feeds providing some uniqueness in content, for example the ASDI data feed. Additional examples of specific interface data feeds are RVR and TFMDI. This level of the schema design represents the external client/vendor interfaces for both incoming and outgoing data. The lowest level of the schema design hierarchy contains commons elements, messages and types shared by higher level schemas, such as ASDI.

As the TFM-M development effort progresses other parts of the TFM-M schema will be utilized, however for the purpose of this ICD the focus will be on the elements, attributes and types required for the initial release of the ETMS ASDI XML data feed.

Figure1 illustrates the hierarchal schema design used to develop the XML schemas for TFM-M and the ASDI XML data feed.

**Figure 6-1. TFM-M XML Schema Design**



### 6.1.1 Required ASDI Schema Definition Files and Locations

The XML developed for the ASDI data feed is defined by XML Schema Definition (XSD) files. These XSD files are used to describe and validate the elements used in the XML documents sent over the ASDI data feed. The following schema definition files are needed to validate, parse and process ASDI XML documents, they are;

1. **TFMS\_XIS.xsd** – This is the root TFMS schema definition which includes all external interface schemas using the same name space. For the initial release ASDI is the only external interface schema that is available and included.
2. **ASDI.xsd** – This is the schema definition used to represent all inbound and outbound data transmitted over the ASDI interface. Only the outbound portion of the schema is used for the initial release of the ASDI XML data feed.
3. **MessageMetaData.xsd** – This is the schema definition used for all TFMS external interface data to represent common header type information.
4. **NasXCommonMessages.xsd** – This schema definition describes common messages sent and received over multiple TFMS interfaces such as ASDI. These messages are built from the data types defined in the NasXCoreElements schema which make use of standard XML schema data types defined by the World Wide Web Consortium (W3C).
5. **NasXCoreElements.xsd** – This schema definition describes all common element types used by NasXCommonMessages.xsd.

These XSD files were developed as part of the TFM-M program and as such contain additional element/attribute definitions supporting multiple TFM-M data interfaces and in some cases data is represented that will be available in future TFM-M releases. Appendix B provides a reference to identify the elements/attributes not used in the initial ETMS ASDI XML data feed.

The ASDI XML documents formed using these schema definitions are contained in the compressed payload portion of the ASDI data packet, this compressed payload will contain multiple batched messages as discussed in prior sections.

## 6.2 Structure of an ASDI XML Document

As discussed in a prior section an ASDI data packet contains two parts, an uncompressed fixed length ASCII header and a variable length compressed “payload” of data. The ASCII header is described in Section 5. The compressed payload portion of the ASDI data packet contains the XML formatted messages. This portion of the data packet must be uncompressed prior to processing ASDI XML messages.



Once uncompressed, the XML payload is comprised of a document preamble followed by multiple ASDI XML formatted messages. The number of messages following the preamble is determined by a configuration value established prior to activating the ASDI feed.

### 6.2.1 ASDI XML Preamble

The preamble included at the beginning of every ASDI XML document lists the namespaces used in forming the ASDI messages contained in the document. The following is an example of the ASDI XML preamble segment of the document.

```
<?xml version="1.0"?>
<asdiOutput xmlns="http://tfm.faa.gov/tfms/TFMS_XIS"
xmlns:nxce="http://tfm.faa.gov/tfms/NasXCoreElements"
xmlns:mmd="http://tfm.faa.gov/tfms/MessageMetaData"
xmlns:nxcm="http://tfm.faa.gov/tfms/NasXCommonMessages"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tfm.faa.gov/tfms/TFMS_XIS
http://www.fly.faa.gov/ASDI/asdidocs/TFMS_XIS.xsd"
timestamp="2007-02-05T21:02:24.0Z">
```

### 6.2.2 ASDI XML Messages

Appendix A provides actual sample ASDI messages in the current format and the corresponding XML format. These sample messages are provided as reference to help guide vendors/users with writing software to parse and validate the ASDI XML documents.

A larger sampling of ASDI XML documents is available at:

<http://www.fly.faa.gov/ASDI/asdi.html>

## 6.3 Development, Test and Validation tools

The XML and XSD files developed for the ASDI data feed adhere to standards and recommendations defined by the W3C, any XML development tool based on these standards and recommendations can be used for developing software to process ASDI XML documents and messages.

However the recommended tool for developing and validating ASDI XML documents and XSD schema definitions is Altova® XMLSpy®. The version currently in use by the ETMS development team is; Enterprise Edition version 2006 rel. 3 sp1.

Additionally, the Java programming language provides a number of built-in Java classes that can be used by ASDI clients to validate an ASDI XML document from within Java source code. An example of how to use these Java classes for validation is available at:

<http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/package-summary.html>

There are two levels of ASDI XML document validation. The first level of validation is the determination as to whether or not there are any errors in the raw NAS input received by the ASDI XML process. If such an error is detected, an error will be logged and the message will not be sent out in the ASDI XML feed. For example, suppose a NAS message is received containing an invalid value for MACH speed, as specified by the schema. In the case the ASDI XML process would detect this error, log the message and not send it out in the feed.

The second level of validation is the detection of any errors relative to the construction and format of the document. This level of validation ensures that the document was constructed in the format defined in the various XSD schema definition files. An ASDI client should never receive an ASDI XML document that fails this level of validation as that would imply a software bug in the ASDI distribution software that constructed the document.

## 6.4 Hints to Vendors to help parse ASDI XML documents

The Java programming language provides a built-in Java class (the SAXParser) that ASDI clients can utilize from within Java source code to parse individual NAS data elements from an ASDI XML document. The SAXParser API can be found at:

<http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/parsers/SAXParser.html>

And a SAXParser tutorial is available at:

<http://java.sun.com/webservices/jaxp/dist/1.1/docs/tutorial/sax/index.html>

The Java programming language also provides a framework (JAXB) that enables a user to create a Java object from an XML document resulting in a Java object representation of the data contained in the XML document. JAXB information is available at:

<http://java.sun.com/webservices/jaxb/>

A freely available C language source library that provides functionality similar to the Java SAXParser is available at:

<http://directory.fsf.org/expat.html>

A freely available C++ language source library that provides XML validation and parsing functionality is available at:

<http://xml.apache.org/xerces-c/>

## Appendix A Sample ASDI Messages

---

This section contains one example of each type of message in the legacy format. The next section shows these same messages in the new XML format. This allows the reader to see the relationship between the old format and the new.

### A.1 ASDI Messages in the Original Format

22215942KZMEAF AAL1765 STL DAL 06 3642N/09252W 07 E2159 10  
STL./.RZC062066..FINGR.FINGR3.DAL

22215943KZMETZ N51CT/565 197 220C 3446N/08502W

22215952KZMEFZ SWA158/913 B733/J 0430 BNA P2356 300  
BNA..BNA.J46.ARG..SGF.TYGER5.MCI/0127

22215954KZMEUZ TRS145 B712/Q 0453 3718N/08610W E2206 370  
DTW./.IIU..BWG.BWGTRANS.RMG.ERLIN2.ATL/2252

22215958KZOABZ SWA1953/883 BUR SJC 2046

22215945KZNYDZ EGF875/069 E135/Q LGA D2200 CLE 2345

22215938ETMSRT EVA607 897 1YO KBG20 X.G20G20AJC S0Q442> /91(@  
J& WJLAX RCTPL9 V;0:Z X:0=> Z50>| / 0?9 /&0?' +I0]P +X0]% !!10) )L1AT  
;D1K9 =91T: >O1" { =!1'5 <<DD: :RDK? -HDN. )(DR- )JDS[ '=DWB 'PDXA  
|3D+G +ND%/ Z/D,7 YYD;C XJD> WZD>T VMD?. NME8Q  
NGE8%ZLALAXVTU LAX LIBBO BRINY BOARS AMAKR LINUZ NATTE  
ZANNG 51140 55150 OGGOE OFORD ONEIL OPAKE OLCOT OPHET OGDEN  
OMOTO OPULO ONEMU OATIS GOC NAKTU XMC KEC APU V299  
G469 R591 R341 R580  
LOSE>KLAX./.RZS122061..VTU200020..LIBBO..BRINY..BOARS..AMAKR..LI  
NUZ..NATTE..ZANNG..5100N/14000W..5500N/15000W..5700N/16000W..OGGO  
E..OFORD..ONEIL..OPAKE..OLCOT..OPHET..OGDEN..OMOTO.R580.OATIS.O  
TR3.GOC.W18.NAKTU.V58.XMC.V52.KEC.A1.APU.AU1A.RCTP/1220

22215948KOOATO UAL893 471 22/2200 320 4019N/12702W 22/2221 320  
4200N/13000W 22/2329 320 4700N/14000W KSFO RKSJ

22215947KZJXAZ PDT4212 CLT HXD A2200

22215957KZAURZ N1017L/072 MSN IMT

## A.2 ASDI Messages in XML Format

This subsection contains an uncompressed payload from one XML data packet that contains the preamble to a packet of messages and then contains one sample of each type of message. For a much longer collection of sample messages, refer to the file on the website given in Section 1.3. The messages below match those given in the legacy format in section A.1.

```
<?xml version="1.0"?>
<asdiOutput xmlns="http://tfm.faa.gov/tfms/TFMS_XIS"
xmlns:nxce="http://tfm.faa.gov/tfms/NasXCoreElements"
xmlns:mmd="http://tfm.faa.gov/tfms/MessageMetaData"
xmlns:nxcm="http://tfm.faa.gov/tfms/NasXCommonMessages"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tfm.faa.gov/tfms/TFMS_XIS TFMS_XIS.xsd"
timestamp="2007-02-08T20:28:49.0Z">
  <asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:42.0Z">
    <flightPlanAmendmentInformation>
      <nxcm:qualifiedAircraftId>
        <nxce:aircraftId>AAL1765</nxce:aircraftId>
        <nxce:departurePoint>
          <nxce:fix>
            <nxce:namedFix>STL</nxce:namedFix>
          </nxce:fix>
        </nxce:departurePoint>
        <nxce:arrivalPoint>
          <nxce:fix>
            <nxce:namedFix>DAL</nxce:namedFix>
          </nxce:fix>
        </nxce:arrivalPoint>
      </nxcm:qualifiedAircraftId>
      <nxcm:amendmentData>
        <nxcm:newCoordinationPoint>
          <nxce:latLong>
            <nxce:latitude>
              <nxce:latitudeDMS degrees="36" minutes="42" direction="NORTH"/>
            </nxce:latitude>
            <nxce:longitude>
              <nxce:longitudeDMS degrees="092" minutes="52" direction="WEST"/>
            </nxce:longitude>
          </nxce:latLong>
        </nxcm:newCoordinationPoint>
        <nxcm:newCoordinationTime type="ESTIMATED">2007-02-
08T21:59:00.0Z</nxcm:newCoordinationTime>
        <nxcm:newRouteOfFlight
legacyFormat="STL../RZC062066..FINGR.FINGR3.DAL"/>
      </nxcm:amendmentData>
    </flightPlanAmendmentInformation>
  </asdiMessage>
</asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:43.0Z">
```

```
<trackInformation>
  <nxcn:aircraftId>N51CT</nxcn:aircraftId>
  <nxcn:computerId>
    <nxce:idNumber>565</nxce:idNumber>
  </nxcn:computerId>
  <nxcn:speed>197</nxcn:speed>
  <nxcn:reportedAltitude>
    <nxce:assignedAltitude>
      <nxce:simpleAltitude>220C</nxce:simpleAltitude>
    </nxce:assignedAltitude>
  </nxcn:reportedAltitude>
  <nxcn:position>
    <nxce:latitude>
      <nxce:latitudeDMS degrees="34" minutes="46" direction="NORTH"/>
    </nxce:latitude>
    <nxce:longitude>
      <nxce:longitudeDMS degrees="085" minutes="02" direction="WEST"/>
    </nxce:longitude>
  </nxcn:position>
</trackInformation>
</asdiMessage>
<asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:52.0Z">
  <flightPlanInformation>
    <nxcn:aircraftId>SWA158</nxcn:aircraftId>
    <nxcn:computerId>
      <nxce:idNumber>913</nxce:idNumber>
    </nxcn:computerId>
    <nxcn:flightAircraftSpecs equipmentQualifier="J">B733</nxcn:flightAircraftSpecs>
    <nxcn:speed>
      <nxce:filedTrueAirSpeed>0430</nxce:filedTrueAirSpeed>
    </nxcn:speed>
    <nxcn:coordinationPoint>
      <nxce:namedFix>BNA</nxce:namedFix>
    </nxcn:coordinationPoint>
    <nxcn:coordinationTime type="PROPOSED">2007-02-
08T23:56:00.0Z</nxcn:coordinationTime>
    <nxcn:altitude>
      <nxce:assignedAltitude>
        <nxce:simpleAltitude>300</nxce:simpleAltitude>
      </nxce:assignedAltitude>
    </nxcn:altitude>
    <nxcn:routeOfFlight
legacyFormat="BNA..BNA.J46.ARG..SGF.TYGER5.MCI/0127"/>
  </flightPlanInformation>
</asdiMessage>
<asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:54.0Z">
  <boundaryCrossingUpdate>
    <nxcn:aircraftId>TRS145</nxcn:aircraftId>
    <nxcn:flightAircraftSpecs
equipmentQualifier="Q">B712</nxcn:flightAircraftSpecs>
    <nxcn:speed>
```

```

    <nxce:filedTrueAirSpeed>0453</nxce:filedTrueAirSpeed>
  </nxcm:speed>
  <nxcm:boundaryPosition boundaryCrossingTime="2007-02-08T22:06:00.0Z">
    <nxce:latitude>
      <nxce:latitudeDMS degrees="37" minutes="18" direction="NORTH"/>
    </nxce:latitude>
    <nxce:longitude>
      <nxce:longitudeDMS degrees="086" minutes="10" direction="WEST"/>
    </nxce:longitude>
  </nxcm:boundaryPosition>
  <nxcm:reportedAltitude>
    <nxce:assignedAltitude>
      <nxce:simpleAltitude>370</nxce:simpleAltitude>
    </nxce:assignedAltitude>
  </nxcm:reportedAltitude>
  <nxcm:routeOfFlight
legacyFormat="DTW./.IIU..BWG.BWGTRANS.RMG.ERLIN2.ATL/2252"/>
  </boundaryCrossingUpdate>
</asdiMessage>
<asdiMessage sourceFacility="KZOA" sourceTimeStamp="2007-01-22T21:59:58.0Z">
  <beaconCodeInformation>
    <nxcm:qualifiedAircraftId>
      <nxce:aircraftId>SWA1953</nxce:aircraftId>
      <nxce:computerId>
        <nxce:idNumber>883</nxce:idNumber>
      </nxce:computerId>
      <nxce:departurePoint>
        <nxce:fix>
          <nxce:namedFix>BUR</nxce:namedFix>
        </nxce:fix>
      </nxce:departurePoint>
      <nxce:arrivalPoint>
        <nxce:fix>
          <nxce:namedFix>SJC</nxce:namedFix>
        </nxce:fix>
      </nxce:arrivalPoint>
    </nxcm:qualifiedAircraftId>
    <nxcm:beaconCode>2046</nxcm:beaconCode>
  </beaconCodeInformation>
</asdiMessage>
<asdiMessage sourceFacility="KZNY" sourceTimeStamp="2007-01-22T21:59:45.0Z">
  <departureInformation>
    <nxcm:qualifiedAircraftId>
      <nxce:aircraftId>EGF875</nxce:aircraftId>
      <nxce:computerId>
        <nxce:idNumber>069</nxce:idNumber>
      </nxce:computerId>
      <nxce:departurePoint>
        <nxce:fix>
          <nxce:namedFix>LGA</nxce:namedFix>
        </nxce:fix>
      </nxce:departurePoint>
    </nxcm:qualifiedAircraftId>
  </departureInformation>
</asdiMessage>
```

```
</nxce:departurePoint>
<nxce:arrivalPoint>
  <nxce:fix>
    <nxce:namedFix>CLE</nxce:namedFix>
  </nxce:fix>
</nxce:arrivalPoint>
</nxcm:qualifiedAircraftId>
<nxcm:flightAircraftSpecs
equipmentQualifier="Q">E135</nxcm:flightAircraftSpecs>
  <nxcm:timeOfDeparture estimated="false">2007-02-
08T22:00:00.0Z</nxcm:timeOfDeparture>
  <nxcm:timeOfArrival estimated="true">2007-02-
08T23:45:00.0Z</nxcm:timeOfArrival>
</departureInformation>
</asdiMessage>
<asdiMessage sourceFacility="ETMS" sourceTimeStamp="2007-01-22T21:59:38.0Z"
trigger="FA">
  <flightManagementInformation>
    <nxcm:qualifiedAircraftId>
      <nxce:aircraftId>EVA607</nxce:aircraftId>
      <nxce:computerId>
        <nxce:idNumber>897</nxce:idNumber>
      </nxce:computerId>
      <nxce:departurePoint facilityId="ZLA">
        <nxce:airport>LAX</nxce:airport>
      </nxce:departurePoint>
      <nxce:arrivalPoint>
        <nxce:airport>RCTP</nxce:airport>
      </nxce:arrivalPoint>
    </nxcm:qualifiedAircraftId>
    <nxcm:aircraftInformation>
      <nxce:flightStatus>ACTIVE</nxce:flightStatus>
      <nxce:aircraftCategory>JET</nxce:aircraftCategory>
      <nxce:userCategory>COMMERCIAL</nxce:userCategory>
    </nxcm:aircraftInformation>
    <nxcm:flightTimeData estimatedDeparture="2007-01-22T21:54:00.0Z"
estimatedArrival="2007-01-23T11:46:00.0Z" originalDeparture="2007-01-22T21:26:00.0Z"
originalArrival="2007-01-23T10:26:00.0Z"/>
    <nxcm:flightTraversalData2>
      <nxce:fix sequenceNumber="1">VTU</nxce:fix>
      <nxce:fix sequenceNumber="2">LAX</nxce:fix>
      <nxce:fix sequenceNumber="3">LIBBO</nxce:fix>
      <nxce:fix sequenceNumber="4">BRINY</nxce:fix>
      <nxce:fix sequenceNumber="5">BOARS</nxce:fix>
      <nxce:fix sequenceNumber="6">AMAKR</nxce:fix>
      <nxce:fix sequenceNumber="7">LINUZ</nxce:fix>
      <nxce:fix sequenceNumber="8">NATTE</nxce:fix>
      <nxce:fix sequenceNumber="9">ZANNG</nxce:fix>
      <nxce:fix sequenceNumber="10">51140</nxce:fix>
      <nxce:fix sequenceNumber="11">55150</nxce:fix>
      <nxce:fix sequenceNumber="12">OGGOE</nxce:fix>
```

```
<nxce:fix sequenceNumber="13">OFORD</nxce:fix>
<nxce:fix sequenceNumber="14">ONEIL</nxce:fix>
<nxce:fix sequenceNumber="15">OPAKE</nxce:fix>
<nxce:fix sequenceNumber="16">OLCOT</nxce:fix>
<nxce:fix sequenceNumber="17">OPHET</nxce:fix>
<nxce:fix sequenceNumber="18">OGDEN</nxce:fix>
<nxce:fix sequenceNumber="19">OMOTO</nxce:fix>
<nxce:fix sequenceNumber="20">OPULO</nxce:fix>
<nxce:fix sequenceNumber="21">ONEMU</nxce:fix>
<nxce:fix sequenceNumber="22">OATIS</nxce:fix>
<nxce:fix sequenceNumber="23">GOC</nxce:fix>
<nxce:fix sequenceNumber="24">NAKTU</nxce:fix>
<nxce:fix sequenceNumber="25">XMC</nxce:fix>
<nxce:fix sequenceNumber="26">KEC</nxce:fix>
<nxce:fix sequenceNumber="27">APU</nxce:fix>
<nxce:waypoint sequenceNumber="1" latitudeDecimal="2037"
longitudeDecimal="7104"/>
<nxce:waypoint sequenceNumber="2" latitudeDecimal="2160"
longitudeDecimal="7310"/>
<nxce:waypoint sequenceNumber="3" latitudeDecimal="2238"
longitudeDecimal="7360"/>
<nxce:waypoint sequenceNumber="4" latitudeDecimal="2294"
longitudeDecimal="7388"/>
<nxce:waypoint sequenceNumber="5" latitudeDecimal="2340"
longitudeDecimal="7425"/>
<nxce:waypoint sequenceNumber="6" latitudeDecimal="2499"
longitudeDecimal="7590"/>
<nxce:waypoint sequenceNumber="7" latitudeDecimal="2514"
longitudeDecimal="7609"/>
<nxce:waypoint sequenceNumber="8" latitudeDecimal="2646"
longitudeDecimal="7799"/>
<nxce:waypoint sequenceNumber="9" latitudeDecimal="3060"
longitudeDecimal="8400"/>
<nxce:waypoint sequenceNumber="10" latitudeDecimal="3300"
longitudeDecimal="9000"/>
<nxce:waypoint sequenceNumber="11" latitudeDecimal="3420"
longitudeDecimal="9600"/>
<nxce:waypoint sequenceNumber="12" latitudeDecimal="3497"
longitudeDecimal="10415"/>
<nxce:waypoint sequenceNumber="13" latitudeDecimal="3452"
longitudeDecimal="10608"/>
<nxce:waypoint sequenceNumber="14" latitudeDecimal="3402"
longitudeDecimal="-10800"/>
<nxce:waypoint sequenceNumber="15" latitudeDecimal="3252"
longitudeDecimal="-10361"/>
<nxce:waypoint sequenceNumber="16" latitudeDecimal="3180"
longitudeDecimal="-10194"/>
<nxce:waypoint sequenceNumber="17" latitudeDecimal="3086"
longitudeDecimal="-9933"/>
<nxce:waypoint sequenceNumber="18" latitudeDecimal="3058"
longitudeDecimal="-9863"/>
```



```

        <nxce:waypoint sequenceNumber="19" latitudeDecimal="2969"
longitudeDecimal="-9662"/>
        <nxce:waypoint sequenceNumber="20" latitudeDecimal="2940"
longitudeDecimal="-9601"/>
        <nxce:waypoint sequenceNumber="21" latitudeDecimal="2732"
longitudeDecimal="-9223"/>
        <nxce:waypoint sequenceNumber="22" latitudeDecimal="2504"
longitudeDecimal="-8893"/>
        <nxce:waypoint sequenceNumber="23" latitudeDecimal="2269"
longitudeDecimal="-8612"/>
        <nxce:waypoint sequenceNumber="24" latitudeDecimal="2205"
longitudeDecimal="-8421"/>
        <nxce:waypoint sequenceNumber="25" latitudeDecimal="2128"
longitudeDecimal="-8248"/>
        <nxce:waypoint sequenceNumber="26" latitudeDecimal="2082"
longitudeDecimal="-8218"/>
        <nxce:waypoint sequenceNumber="27" latitudeDecimal="2007"
longitudeDecimal="-8148"/>
        <nxce:waypoint sequenceNumber="28" latitudeDecimal="1511"
longitudeDecimal="-7291"/>
        <nxce:waypoint sequenceNumber="29" latitudeDecimal="1505"
longitudeDecimal="-7273"/>
        <nxce:airway sequenceNumber="1">V299</nxce:airway>
        <nxce:airway sequenceNumber="2">G469</nxce:airway>
        <nxce:airway sequenceNumber="3">R591</nxce:airway>
        <nxce:airway sequenceNumber="4">R341</nxce:airway>
        <nxce:airway sequenceNumber="5">R580</nxce:airway>
        <nxce:center sequenceNumber="1">ZLA</nxce:center>
        <nxce:center sequenceNumber="2">ZOA</nxce:center>
        <nxce:center sequenceNumber="3">ZSE</nxce:center>
        <nxce:center sequenceNumber="4">ZAN</nxce:center>
        <nxce:center sequenceNumber="5">ZPA</nxce:center>
        <nxce:sector sequenceNumber="1">ZLALAX</nxce:sector>
    </nxcm:flightTraversalData2>
    <nxcm:routeOfFlight
legacyFormat="KLAX../RZS122061..VTU200020..LIBBO..BRINY..BOARS..AMAKR..LI
NUZ..NATTE..ZANNG..5100N/14000W..5500N/15000W..5700N/16000W..OGGOE..OFO
RD..ONEIL..OPAKE..OLCOT..OPHET..OGDEN..OMOTO.R580.OATIS.OTR3.GOC.W1
8.NAKTU.V58.XMC.V52.KEC.A1.APU.AU1A.RCTP/1220"/>
    </flightManagementInformation>
</asdiMessage>
<asdiMessage sourceFacility="KOOA" sourceTimeStamp="2007-01-22T21:59:48.0Z">
    <oceanicReport>
        <nxcm:qualifiedAircraftId>
            <nxce:aircraftId>UAL893</nxce:aircraftId>
            <nxce:departurePoint>
                <nxce:airport>KSFO</nxce:airport>
            </nxce:departurePoint>
            <nxce:arrivalPoint>
                <nxce:airport>RKSI</nxce:airport>
            </nxce:arrivalPoint>
        </nxcm:qualifiedAircraftId>
    </oceanicReport>
</asdiMessage>

```

```
</nxcm:qualifiedAircraftId>
<nxcm:speed>471</nxcm:speed>
<nxcm:plannedPositionData planNumber="1">
  <nxce:position>
    <nxce:latitude>
      <nxce:latitudeDMS degrees="42" minutes="00" direction="NORTH"/>
    </nxce:latitude>
    <nxce:longitude>
      <nxce:longitudeDMS degrees="130" minutes="00" direction="WEST"/>
    </nxce:longitude>
  </nxce:position>
  <nxce:altitude>320</nxce:altitude>
  <nxce:time>2007-01-22T22:21:00.0Z</nxce:time>
</nxcm:plannedPositionData>
<nxcm:plannedPositionData planNumber="2">
  <nxce:position>
    <nxce:latitude>
      <nxce:latitudeDMS degrees="47" minutes="00" direction="NORTH"/>
    </nxce:latitude>
    <nxce:longitude>
      <nxce:longitudeDMS degrees="140" minutes="00" direction="WEST"/>
    </nxce:longitude>
  </nxce:position>
  <nxce:altitude>320</nxce:altitude>
  <nxce:time>2007-01-22T23:29:00.0Z</nxce:time>
</nxcm:plannedPositionData>
<nxcm:reportedPositionData>
  <nxce:position>
    <nxce:latitude>
      <nxce:latitudeDMS degrees="40" minutes="19" direction="NORTH"/>
    </nxce:latitude>
    <nxce:longitude>
      <nxce:longitudeDMS degrees="127" minutes="02" direction="WEST"/>
    </nxce:longitude>
  </nxce:position>
  <nxce:altitude>320</nxce:altitude>
  <nxce:time>2007-01-22T22:00:00.0Z</nxce:time>
</nxcm:reportedPositionData>
</oceanicReport>
</asdiMessage>
<asdiMessage sourceFacility="KZJX" sourceTimeStamp="2007-01-22T21:59:47.0Z">
  <arrivalInformation>
    <nxcm:qualifiedAircraftId>
      <nxce:aircraftId>PDT4212</nxce:aircraftId>
    <nxce:departurePoint>
      <nxce:fix>
        <nxce:namedFix>CLT</nxce:namedFix>
      </nxce:fix>
    </nxce:departurePoint>
    <nxce:arrivalPoint>
      <nxce:fix>
```

```
        <nxce:namedFix>HXD</nxce:namedFix>
      </nxce:fix>
    </nxce:arrivalPoint>
  </nxcm:qualifiedAircraftId>
  <nxcm:timeOfArrival estimated="false">2007-02-
08T22:00:00.0Z</nxcm:timeOfArrival>
</arrivalInformation>
</asdiMessage>
<asdiMessage sourceFacility="KZAU" sourceTimeStamp="2007-01-22T21:59:57.0Z">
  <flightPlanCancellation>
    <nxcm:qualifiedAircraftId>
      <nxce:aircraftId>N1017L</nxce:aircraftId>
      <nxce:computerId>
        <nxce:idNumber>072</nxce:idNumber>
      </nxce:computerId>
      <nxce:departurePoint>
        <nxce:fix>
          <nxce:namedFix>MSN</nxce:namedFix>
        </nxce:fix>
      </nxce:departurePoint>
      <nxce:arrivalPoint>
        <nxce:fix>
          <nxce:namedFix>IMT</nxce:namedFix>
        </nxce:fix>
      </nxce:arrivalPoint>
    </nxcm:qualifiedAircraftId>
  </flightPlanCancellation>
</asdiMessage>
</asdiOutput>
```

## Appendix B ASDI Schema Elements

### B.1 Table Descriptions

For each XSD noted above in Section 6, there is a corresponding table if that schema has any element/attribute that is either not used in the initial versions or if the definition of the element/attribute is a superset of what will actually be included in the initial versions of the ASDI XML feed. This appendix is provided as guide to vendor/users to identify areas that do not need to be coded for in the initial release. Following describes the various columns of the tables:

- **Element/Attribute** – Identifies the element/attribute.
- **Description** - Provides a high level description of meaning of the specified element/attribute. The content is largely derived from the annotation in the corresponding schema for the given element or attribute.
- **Status** - Indicates whether the specified element/attribute is not used nor has a superset definition. If the latter case, then the Status field also qualifies the superset definition.
- **Comments** - Provided for any optional comment/feedback.

#### B.1.1 ASDI.xsd

ASDI.xsd is the top-level schema that describes the overall organization of the XML data. The following table enumerates those components of this schema that either will not be in the initial versions of the ETMS ASDI XML Server, or qualifies the representation of the element/attribute.

Element/Attribute	Description	Status	Comments
AsdiInput	Root node for all elements received by TFMS.	Not used in the current version	
AsdiOutput	Root node for all elements sent by TFMS to ASDI clients.	Only the <i>timestamp</i> attribute of this element will be present. None of the other attributes of this element will be included.	
triggerType	Attribute of the <i>asdiMessage</i> element. Simple enumerated type used to describe what triggered an ASDI message or event.	Only included for message type <i>flightInformationMessageType</i> (i.e. RT messages). Will never be present for any other message type.	

#### B.1.2 NasCommon Elements.xsd

NasCommonElements.xsd is the schema that contains the high-level element descriptions; such as the description of all the fields in a given message type. The following table enumerates those components of this schema that either will not be in the initial versions of the ASDI XML Server, or qualifies the representation of the element/attribute.

Element/Attribute	Description	Status	Comments
FlightPlanInformation	Complex type describing the components of a FlightPlanInformation element (represents a FZ message).	The optional child element, <i>BeaconCode</i> is never used .	

### B.1.3 NasCoreElements.xsd

NasCoreElements.xsd is the schema that contains the majority of the detailed element descriptions. The following table enumerates those components of this schema that either will not be in the initial versions of the ASDI XML Server, or qualifies the representation of the element/attribute.

Element/Attribute	Description	Status	Comments
adaptedRouteSegment	Simple type describing part of a flight's adapted route consisting of two fixes and the route between them.	Not used in the current version	
AirlineIdType	Simple type defining 3 uppercase letters that prefix an airline designation.	Not used in the current version	
airwayNameType	Simple type defining a valid format of 2-8 alphanumerics	Not used in the current version	
assignedAltitudeType	Complex type defining the altitude assigned to this flight by Air Traffic Control.	If the child element of this element is <i>simpleAltitude</i> and the optional attribute, <i>above</i> is defined, it will always be set to "true".	
codedRouteNameType	Simple type defining a valid format of 2 to 8 alphanumerics with the exception of "XXX", "VFR" and "DVFR".	Not used in the current version	
computerSystemIdType	Simple enumerated type used to predefine the identification of systems which the TFMS exchanges data with.	Not used in the current version	
coordinationTimeType		Optional attribute, <i>delayTime</i> , will not be used	
delayDataType	Simple type defining delay data from field 10 in the format of "/D(d)d+dd where the first two digits(hours) must not exceed 21 and the last two digits(minutes) must not exceed 59	Not used in the current version	
distanceType	Simple type defining a distance in the range of 0 thru 999	Not used in the current version	
enRouteElementType	Complex type defining an enRoute element	Not used in the current version	

Element/Attribute	Description	Status	Comments
flightClass	Simple enumerated type defining the flight class of the aircraft which is determined by the aircraft call sign that is in the ASDI feed.	Not used in the current version	
flightComputerId	Complex type used to uniquely identify a flight across systems.	This element will only be present if the CID of a flight is in the message. The only child element currently used is the <i>idNumber</i> , which represents the CID. The other two optional child elements, <i>facilityIdentifier</i> and <i>computerSystemId</i> , are not currently used.	
flightPlanRemarks	Complex type defining comments related to a flight plan	Not used in the current version	
flightTraversalData	Complex type defining a collection of airspace the referent flight traverses. Within each unit of airspace, the fixes the flight traverses as well as the position data for those fixes. If the route of flight contains an adapted route, the route name is given, and those fixes which make up the segment of the route in the given airspace is defined.	Not used in the current version	
fpaNumberType	Simple type used to identify an FPA number	Not used in the current version	
hoursMinutesTimeType	Simple type describing time format currently used in ASDI fields	Not used in the current version	
latitudeType	Complex type	Only <i>latitudeDMS</i> is currently used. <i>latitudeDecimal</i> and <i>latitudeRadians</i> representations are not currently used.	
longitudeType	Complex type	Only <i>longitudeDMS</i> is currently used. <i>longitudeDecimal</i> and <i>longitudeRadians</i> representations are not currently used.	
loopReEntryType	Simple type defining a coded route with a re-entry option adapted; which includes an adapted entry/exit fix.	Not used in the current version	

Element/Attribute	Description	Status	Comments
qualifiedAircraftIdType	Complex type defining the combination of elements contained within a flight plan, which together can be used to more accurately identify a particular flight.	<p>The optional attribute, <i>flightClass</i>, is not currently used.</p> <p>The optional attribute, <i>facilityId</i>, of the child element arrivalPoint is not currently used.</p> <p>The optional attribute, <i>facilityId</i>, of the child element departurePoint may only be present in the flightManagementInformationType (i.e. RT) message.</p> <p>The optional child element, <i>fix</i>, of the child element arrivalPoint is not currently used in an oceanicReportType message (i.e. TO).</p> <p>The optional child element, <i>fix</i>, of the child element departurePoint is not currently used in an oceanicReportType message (i.e. TO) or a flightManagementInformationType (i.e. RT) message.</p> <p>The optional child element, <i>airport</i>, of the child element arrivalPoint may only be present in an oceanicReportType (i.e. TO) or a flightManagementInformationType (i.e. RT) message; it will never be in any other message type.</p> <p>The optional child element, <i>airport</i>, of the child element departurePoint may only be present in an oceanicReportType (i.e. TO) or a flightManagementInformationType (i.e. RT) message; it will never be in any other message type.</p>	
qualifiedAirspaceType	Complex type defining the structure and constraints for describing a qualified airspace.	Not used in the current version	

Element/Attribute	Description	Status	Comments
reportedAltitudeType	Complex type defining the structure for describing the elements necessary to provide a flight's reported altitude.	Not used in the current version	
requestedAltitudeType	Complex type defining the structure for describing the elements necessary to provide a flight's requested altitude.	If the child element of this element is <i>simpleAltitude</i> and the optional attribute, <i>above</i> is defined, it will always be set to "true".	
routeOfFlightType	Complex type defining the structure for describing a flight's route of flight.	Only the attribute, <i>legacyFormat</i> , will be present. None of the other attributes or child elements are currently used.	
routeTypesType	Simple enumerated type defining the structure for describing the elements necessary to define the identity of a flight's route type.	Not used in the current version	
sectorNumberType	Simple type used to place constraints on the sector identification number.	Not used in the current version	
simpleAltitudeType		If present, the optional attribute, <i>above</i> , will always be set to "true".	
standardArrivalRouteNameType	Simple type used to place constraints on a flight's standard arrival route name.	Not used in the current version	
standardInstrumentDepartureType	Complex type defining the structure for describing a standard Instrument Departure (SID) type.	Not used in the current version	
standardInstrumentDepartureRouteNameType	Simple type defining the structure and constraints for a standard Instrument Departure (SID) Route Name type.	Not used in the current version	
visualFlightRulesType	Complex type defining the structure for describing a flight's visual flight rules type.	If present, the optional attribute, <i>vfrOnTop</i> , will always be set to "true".	